# UnlimitID

Marios Isaakidis, Harry Halpin, George Danezis

# UnlimitID: Privacy-Preserving Federated Identity Management using Algebraic MACs

Marios Isaakidis
University College London
m.isaakidis@cs.ucl.ac.uk

Harry Halpin
INRIA
harry.halpin@inria.fr

George Danezis
University College London
g.danezis@ucl.ac.uk

## ABSTRACT

UnlimitID is a method for enhancing the privacy of commodity OAuth and applications such as OpenID Connect, using anonymous attribute-based credentials based on algebraic Message Authentication Codes (aMACs). OAuth is one of the most widely used protocols on the Web, but it exposes each of the requests of a user for data by each relying party (RP) to the identity provider (IdP). Our approach allows for the creation of multiple persistent and unlinkable pseudo-identities and requires no change in the deployed code of relying parties, only in identity providers and the client.

## 1. INTRODUCTION

One common user need is to sign on from a single provider to multiple services (Single Sign On), where each service can demand proof-of-authentication and authorization of the transfer of personal data. OAuth [6] is the primary protocol for transferring *attributes* (user data) between *identity providers* (IdPs) that host user data and *relying parties* (RPs) that demand user data. OAuth is considered to be one of the most popular protocols on the Web, being used by major providers such as Google, Twitter, Github, and government services like *GOV.UK Verify*.

One key privacy shortcoming of OAuth is that it offers no unlinkability between the IdP and the RPs in terms of requests for the identity or data of a user. The IdP is considered completely trusted to provide authoritative information on behalf of a user, and an honest but curious IdP can collect detailed data on user behavior by logging the transactions between itself and RPs on a per-user basis. The RPs also link the user to their IdP, preventing pseudonymous or anonymous usage of their service.

We present UnlimitID, which uses lightweight attribute-based anonymous credentials based on algebraic MACs [1] to provide unlinkability between an IdP and the various RPs for a given user using unlinkable, but multi-use, pseudo-identities. It can be deployed easily today as it requires no changes to OAuth-compliant RPs, but only modest changes to IdPs and the clients, and can enable unlinkable data flows even when the user is offline. Section 2 overviews the OAuth protocol and related literature, Section 3 describes UnlimitID, Section 4 describes the implementation and performance considerations, and Section 5 evaluates the prospect of a large scale deployment.



ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

## 2. BACKGROUND

The protocol flow of OAuth involves a user, an identity provider (IdP), and a relying party (RP) so that the RP can access *attributes* on an IdP via the use of an access token. Once the user authenticates to the IdP, typically via HTTP redirection from the RP, the IdP provides an access token to the user, which when presented by the user to the RP authorizes the server-side transfer of data from the IdP to the RP. OAuth is a generalized protocol and so specific variations may not be interoperable. OpenID Connect is an established adaptation that standardizes OAuth for transferring personal data, given as key-value pairs (such as proof-of-authentication, name, age, photos) [9]. OAuth can be interpreted as a *capabilities*-based system, where the authorized access token effectively acts as a capability to access attributes [5]. It is not necessary to link capabilities to a single identity.

Previous work has discovered a large number of security vulnerabilities in OAuth 2.0, but a more fundamental problem that has not been addressed is that the identity provider in OAuth is assumed to be completely trusted and so may log all user transactions with RPs and possibly even impersonate a user on a RP (the "mixed-up IdP" attack) [4]. For example, if Google OpenID Connect is used to authenticate and authorize file access in an "ephemeral" instant messaging application like Wickr, Google would have metadata about information sent from Google to Wickr for a particular user even if Wickr deleted all metadata and information.

A number of techniques have been proposed for mitigating the privacy risk inherent in the OpenID/OAuth model. One is to change the flow so that data is sent from the IdP to the user and then from the user to the RP, as done by Mozilla BrowserID and SPRESSO [3], achieving unlinkability but at the cost of requiring the client to be online for all transfer of attributes (unusable in practice) and moving the root of trust to the user's browser. Another technique is to use blind signature for proving the possession of an identity at an IdP to a RP without revealing the precise identity, as done by PseudoID [2]. However, PseudoID relies on a cookie for storing the RSA blinded access token so it can only be used for one-time proof-of-authentication rather than the

transfer of attributes. Crypto-Book [8] proposes to enforce privacy by the use of threshold cryptography between multiple identity providers to establish a public key infrastructure (PKI) and then to preserve anonymity vis-a-vis a group using linkable ring signatures to prove group identity and partially blinded signatures for attribute transfer, but the use of threshold cryptography to both establish a PKI and consume credentials via RPs is incompatible with commodity OAuth implementations like OpenID Connect.

# 3. DESIGN OF UNLIMITID

UnlimitID offers the creation of unlinkable *pseudo-identities* that are tied to a RP with the use of multi-show credentials. Our system supports selective attribute disclosure (such as "show age") without the need to reveal the pseudonym, so that attributes can be verified without a persistent pseudo-identity, effectively avoiding the harvesting of attributes over time by the RP or the IdP. In addition, an IdP can allow users to have multiple pseudo-identities with a RP, each of them having different attributes. In this section we analyze the threat model of UnlimitID and the desirable privacy and security properties. Then, we explain how UnlimitID utilizes algebraic MACs to provide unlinkable pseudonyms and how it adapts to the OAuth protocol flow.

## 3.1 Threat Model

The main privacy concern is that an IdP can passively collect the attributes a user selects to disclose to a RP, so the threat model of UnlimitID is against curious but honest IdPs that may independently, or by colluding with RPs, try to uncover what services users are accessing with particular attributes. We can also defend against malicious IdPs that may want to impersonate a user on a RP, i.e. the "mixed-up IdP" attack. Malicious IdPs that deny authenticating a user, the validity of a blinded signature, and the like are out of scope, and the damage would be limited to the refusal of the usage of RP services for a user. We also want to defend against malicious users that take advantage of the anonymous credentials to create multiple pseudonymous accounts with a RP for purposes of a sybil attack.

## 3.2 Privacy and Security Properties

UnlimitID is aiming towards the following properties:

- **Persistent RP accounts ("Consistency" [2])** Users can have long-lived pseudonymous accounts on RPs.
- **Undetectability** IdP and RPs cannot map pseudonyms to the owner of the respective credentials.
- **Unlinkability** IdPs cannot link the different pseudonyms of a user with the same or across RPs.
- **Selective attributes disclosure** Users can choose which subset of their attributes to reveal to the RPs/IdP each time. This property can support fine-grained attribute-based policy systems.
- **Multi-show credentials** Credentials are locally blinded by the user and can be used an arbitrary number of times before their expiration date without revealing the identity of the owner.
- **Unforgability** IdPs cannot impersonate the owner of a pseudo-identity to a RP.
- **Sybil resistance** IdPs can enforce that users may create up to a certain number of pseudo-identities with each RP, with the possibility to limit users to a single pseudonymous account per RP.
- **Non-transferability** An IdP may prevent or discourage the transferability of pseudo-identities across users.

Like OAuth, we do not define the authentication methods or how to securely store and synchronize secrets between devices. We do not control the values of the attributes transferred, and so a user may transfer values that reduce their anonymity set or are uniquely identifying. We assume the confidentiality of tokens in storage and in transit as well as network traffic anonymity (such as provided by anonymous proxy networks) and a secure user environment with trusted cryptographic primitives.

## 3.3 Achieving Unlinkability

UnlimitID uses selective disclosure credentials, based on algebraic messages authentication [1], to achieve its undetectability and unlinkability properties. In particular we use the $\mathsf{MAC}_{GGM}$ construction that is secure in the Generic Group Model.

In a nutshell Unlimited achieves unlinkability in two phases:

1. First, a user authenticates to the IdP and gets issued a credential encoding a long-term secret key and a key/value pair denoting an attribute. The credential also includes an expiry date after which it will need to be re-issued. For the sake of simplicity, we assume that all credentials issues contain four fields: a secret key SUB unknown to the IdP, a KEY and VALUE pair of attributes, and an expiry time EXP.

2. In a second phase, the user blinds the credential and deposits it back to the IdP to generate a pseudo-identity specific to a RP service. At this point the IdP acts in its normal role in the OAuth protocol, using the pseudo-identity and the unlikable attributes to back the authentication, and the user can authenticate to the RP using the standard OAuth flow.

A few more things happen under the hood of this protocol: the issued credential contains a long-term user secret SUB unknown to the IdP, but stable over time for the user. When the user blinds and shows the credential back to the IdP, it uses this secret and specializes it to become a pseudonym for the service that will be requested: the public $Gid$ is derived for a service as $Gid = H(ServiceName)$, and the service specific user pseudonym is derived as $Uid_{\text{Service}} = Gid^{\text{SUB}}$. This pseudonym is persistent for a specific client with a fixed SUB, but unlinkable across services as well to the original issuing protocol given by the IdP. The existence of a fixed, long-term, per-service $Uid_{\text{Service}}$ allows the aggregation of multiple credentials and their attributes into a persistent pseudo-identity for transferring multiple data items or attributes to a RP over multiple OAuth transactions.

### 3.3.1 Issuing of Credentials

The UnlimitID issuing protocol follows closely the issuing protocol of selective disclosure credentials using an algebraic MAC, and we use the notation in Chase et al [1] for the credential protocols. When the IdP is set up it runs the Setup

protocol to enumerate system wide parameters that are public and shared by all system users. It is important to ensure those parameters are used by all, since specializing them to a specific user or a small set of them may allow the IdP to link the two phases of UnlimitID. The IdP also runs the CredKeyGen procedure, to generate a set of public and private parameters, and shares with all its public parameters, which have to be used by all. The private parameters are keys secret to the IdP that are used to issue and verify the blinded message authentication codes – those must be kept secret at all times to preserve authenticity.

Issuing a credential closely follows the standard aMAC based credential BlindIssue as described in Sect. 4.2 of Chase et al. [1], using a single blind attribute for SUB, and three public attributes for KEY, VALUE and EXP. As part of this process the user deposits an encryption of their secret value SUB and proves knowledge of it – this is done once upon account creation and the secret is reused for all subsequent issuance to ensure stable pseudonyms. The result of the issuance is a credential encoding (KEY, VALUE, EXP, SUB) along with an algebraic MAC attesting of its validity. The credential and MAC can be blinded multiple times and used multiple times, across time and sessions with different RPs.

### 3.3.2 Pseudonym registration

Given an issued credential from the IdP, a user performs an unlinkable pseudonym registration as a prelude to an OAuth session. The user runs the Show protocol, proving they have a valid MAC on a credential with known attributes. Besides proving knowledge of the attributes, the (KEY, VALUE, EXP) attributes are revealed and associated with the pseudonym. Furthermore, a service specific pseudonym $\text{Uid}_{\text{SERVICE}}$ is derived as $\text{Uid}_{\text{SERVICE}} = \text{Gid}^{Sub}_{\text{SERVICE}}$ using the user secret SUB, and the service specific $\text{Gid} = H(ServiceName)$, where $H$ is a cryptographically secure hash function.

Technically, the Show protocol outputs four signed Pedersen commitments $(C_{m_0}, C_{m_1}, C_{m_2}, C_{m_3},)$ encoding attributes (KEY, VALUE, EXP, SUB) respectively. We therefore augment the protocol with an additional non-interactive zero-knowledge proof, to prove that the pseudonym is well formed:

$$\begin{aligned}\textbf{NIZK}\{&(\text{SUB}, z_0, z_1, z_2, z_3) : \\ &\text{C}_{\text{m}_0} = \text{h}^{z_0}\text{u}^{\text{KEY}} \wedge \text{C}_{\text{m}_1} = \text{h}^{z_1}\text{u}^{\text{VALUE}} \wedge \\ &\text{C}_{\text{m}_2} = \text{h}^{z_2}\text{u}^{\text{EXP}} \wedge \text{C}_{\text{m}_3} = \text{h}^{z_3}\text{u}^{\text{SUB}} \wedge \\ &\text{Uid}_{\text{SERVICE}} = \text{Gid}^{\text{SUB}}_{\text{SERVICE}}\}\end{aligned}$$

The pseudonym is service specific and unlikable to the originally linked credential, as well as other pseudonyms beyond the anonymity set reduction inherent in revealing the attributes. The stable per-service pseudonym can be used to aggregate multiple attributes as part of a single subsequent OAuth session. Thus multiple credential shows can be performed to endow a pseudonym with multiple attributes. After showing a number of blind credentials and creating a pseudo-identity specific to a service, the traditional OAuth flow can be executed, authenticating the user against the pseudonym rather than their identity at the IdP.

## 3.4 UnlimitID Information Flow

UnlimitID is designed so as to be interoperable with existing OAuth 2.0 protocol deployments, although there is an optional signature verification to prevent the "mixed-up IdP" attack that OAuth2 does not defend against in the standards (mitigations that augment OAuth are under discussion by the OAuth Working Group). In regard to terminology, an UnlimitID user corresponds to an OAuth *Resource Owner*, a RP to a *Client* and an IdP to an *Authorization/Resource Server*.

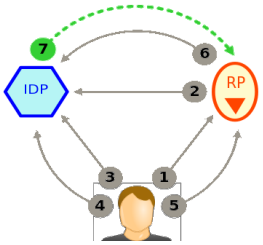


**Figure 1: UnlimitID Information Flow**

### 3.4.1 Protocol Flow

We assume that the user has already been issued a credential for that epoch, as advised in 3.3.1. The flow is presented in Figure 1 and explained in detail in the following steps:

1. A user visits a RP that requires authorization or authentication.
2. The RP redirects the user to the IdP.
3. The user selects which attributes of her credential to reveal, locally blinds it and deposits it along with the aMAC to the IdP's authorization endpoint.
4. The IdP validates the certificate in accordance to the Show protocol in 3.3.2, checks the expiration date and, if still valid, creates a temporary pseudo-identity entry with the revealed attributes in a database. This entry will be stored by the IdP until the expiration of the presented credential. If a pseudo-identity with the same pseudonym already exists, the new attributes are appended and update the respective entry in the database.
5. The IdP redirects the user back to the RP, along with an authorization code for accessing the respective pseudo-identity database entry. In case the pseudonym attribute was revealed, the user appends a signature under the public key $UID_{service} = PK = GID^{Sub}$ on the authorization code, which is kept fresh by using a nonce as per the OAuth standard. Note that the IdP cannot forge the signature, since that would imply it knows *Sub*.
6. The user passes the authorization code to the RP. Optionally to defend against the "mixed-up" IdP attack, when a pseudonym is disclosed the RP will check that the accompanying signature is fresh and that belongs to the owner of the long-term secret key used to derive the claimed pseudonym. Then the RP uses the authorization code to obtain an access token from the IdP's token endpoint.
7. The RP can use the access token to request the pseudo-identity's attributes (e.g. a *UserInfo* structure [9]) from the IdP's resource server endpoint.

### 3.4.2 Credentials rate limiting

The anonymity offered is in general analogous to the amount of active users of the IdP *across epochs* without taking into account details of values transferred, as the precise values in transferred attributes may reduce the anonymity set (for example, by transferring an uniquely identifying attribute). Epochs are created to keep the expiration dates of pseudo-identities uniform so they can not be de-anonymized by virtue of differing expiration dates. Users get new credentials in all epochs as their inactivity could be narrowed to inactive pseudo-identities during that time. In order to prevent timing attacks, we suggest a long expiration window (such as 2 weeks) for credentials. Access tokens cannot exceed the expiration of the respective pseudo-identity; upon expiration, an UnlimitID user can request the re-creation of a pseudo-identity with the same identifier at the IdP. There is a window of time at the end of every epoch when users can request the issuance of new aMACs, which they can use to extend the liveness of their pseudo-identities with the RPs. Note that credentials issued during an epoch are valid till the end of it, even when the respective user account is deleted by the IdP. In the meantime, users may continue presenting the asserted attributes to RP services. As a countermeasure IdPs may enforce own policies at step 4 of the Unlimited protocol flow (see 3.4.1), e.g. check that the user with the revealed email is still active. Anonymous credential blacklisting is a not a goal for UnlimitID, although it could be added. In addition, UnlimitID discourages the sharing of credentials among users, since sharing a pseudo-identity secret subsequently gives access to other pseudo-identities on other RPs derived from that secret.

## 4. PROTOTYPE PERFORMANCE

We implemented the IdP issuing of credentials and pseudonym registration in Python to evaluate its performance.[1] The implementation consists of 1874 lines of Python 3.4 for the IdP credential server, divided into 495 lines for the IdP server, 601 lines of a generic zero-knowledge proof library and 778 lines for the core algebraic MAC protocols, including unit and timing tests. The underlying cryptographic and elliptic curve library used is *petlib* that wraps the OpenSSL *ec* and *bn* modules. The EC curves used throughout is NIST p192 which provides good performance on the x64 Intel architecture thank to the handcrafted constant-time assembler implementation [7]. The server uses the standard library *asyncio* reactor loop framework, limiting computations to a single core.

All tests were performed on an Intel Core i7-4700MQ running at 2.4Ghz with 16.0GB of RAM. The numbers reported are an average of 10 runs. Timings were gathered using the standard library *time.monotonic()* clock and all times are reported in milliseconds (ms). We measured the timing of the full user & server computations for issuing and pseudonym registration, and also separated the user and server components. The issuing protocol takes a total of 38ms seconds per credential, of which 7ms are spend for user computations, and 31ms at the server. The showing protocol takes in total 29ms, of which 15ms are spent on the user and 16ms are spent on the server. The timings reflect what was expected: issuing involves the server performing a blind signature, while the user simply encrypts the secret attribute; whereas, registering the pseudonym involves blinding and generating a zero-knowledge proof at the user, and verifying it at the server – which balances the computational load.

[1] https://github.com/UCL-InfoSec/AnonID_OP

## 5. CONCLUSIONS

UnlimitID demonstrates that the OAuth protocol can be upgraded so as to not violate user privacy by creating unlinkable pseudo-identities based on multi-show attribute-based credentials. Unlike PseudoID [2] and Crypto-Book [8], UnlimitID is usable within current OAuth flows with modifications needed only by the relatively few IdPs, while the multitudes of already-deployed RPs can remain unchanged, except for the optional case where RPs want to protect their users from the "mixed-up IdP" attack by checking that the authorization code is accompanied by a fresh signature under the long-term secret stored by the user that the IdP does not have access to. The anonymity provided by UnlimitID is proportional to the number of active users on the IdP that share the same values in their credentials, and so should increase for large IdPs. As it is compatible with OAuth, it can be deployed and used today, and future work will test UnlimitID with privacy-aware IdPs that want to maintain compatibility with OpenID Connect while maintaining the privacy of their users.

**Acknowledgements.** The authors are supported by NEXT-LEAP (EU H2020 ref: 688722) and EPSRC Grant EP/M013-286/1.

## 6. REFERENCES

[1] M. Chase, S. Meiklejohn, and G. Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 1205–1216, 2014.

[2] A. Dey and S. Weis. Pseudoid: Enhancing privacy in federated login. *HotPETS Workshop*, 2010.

[3] D. Fett, R. Küsters, and G. Schmitz. SPRESSO: A secure, privacy-respecting single sign-on system for the Web. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1358–1369. ACM, 2015.

[4] D. Fett, R. Küsters, and G. Schmitz. A comprehensive formal security analysis of OAuth 2.0. 2016. arXiv preprint arXiv:1601.01229.

[5] H. Halpin and B. Cook. Federated identity as capabilities. In *Annual Privacy Forum*, pages 125–139, 2012.

[6] D. Hardt. The OAuth 2.0 authorization framework, 2012. https://tools.ietf.org/html/rfc6749.

[7] E. Käsper. Fast elliptic curve cryptography in openssl. In *Financial Cryptography and Data Security - FC 2011 Workshops*, pages 27–39, 2011.

[8] J. Maheswaran, D. Jackowitz, E. Zhai, D. I. Wolinsky, and B. Ford. Building privacy-preserving cryptographic credentials from federated online identities. In *Proceedings of the ACM Conference on Data and Application Security and Privacy*, pages 3–13. ACM, 2016.

[9] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore. OpenID Connect Core 1.0, 2014. http://openid.net/specs/openid-connect-core-1_0.html.